To estimate the memory usage of the `CompareDocuments` class as a function of the number of documents n and the dimension d, we need to consider the primary data structures and their sizes:

```
public class CompareDocuments {
    public static void main(String[] args) {
        int k = Integer.parseInt(args[0]);
        int d = Integer.parseInt(args[1]);
        String[] filenames = StdIn.readAllStrings();
        int n = filenames.length;

        // create document sketches
        Sketch[] sketches = new Sketch[n];
        for (int i = 0; i < n; i++) {
            In in = new In(filenames[i]);
            String text = in.readAll();
            sketches[i] = new Sketch(text, k, d);
        }

        // print header
        StdOut.print("    ");
        for (int i = 0; i < n; i++) {
            StdOut.printf("%8.4s", filenames[i]);
        }
        StdOut.println();

        // print n-by-n table
        for (int i = 0; i < n; i++) {
            StdOut.printf("%.4s", filenames[i]);
            for (int j = 0; j < n; j++) {
                StdOut.printf("%8.2f",
sketches[i].similarTo(sketches[j]));
            }
            StdOut.println();
        }
    }
}
```

1. **Array of filenames**:
   - **Memory Usage**: This array will hold references to the filenames.
   - **Size**: n references.
2. **Array of `Sketch` objects**:
   - **Memory Usage**: This array will hold references to the `Sketch` objects.
   - **Size**: n references.
3. **`Sketch` Objects**:

- The `Sketch` class is initialized with the text of the document, and the parameters k and d. Assuming the `Sketch` class processes the text and stores information in an array of size d, the memory usage will depend on the implementation of `Sketch`.

Assuming each `Sketch` object contains:

- A reference to the original document text.
- A data structure (e.g., an array) of size d to store the sketch.

Let's assume each element in the sketch array is a double (8 bytes).

- **Memory Usage** per `Sketch` object:
  - Reference to text: typically 8 bytes (on a 64-bit JVM).
  - Sketch array of size d: $d \times 8$ bytes.

4. **Auxiliary Data Structures**:

- The `String` text data and its internal representation.
- Temporary variables, loop counters, etc.

To sum up, the memory usage can be approximated as follows:

1. **Array of filenames**: $n \times \text{reference\_size}$
   - On a 64-bit JVM, reference size is 8 bytes.
   - Memory: $n \times 8$ bytes.
2. **Array of `Sketch` objects**: $n \times \text{reference\_size}$
   - Memory: $n \times 8$ bytes.
3. **Each `Sketch` object**:
   - Reference to the document text: 8 bytes.
   - Sketch array: $d \times 8$ bytes.
   - Total per `Sketch`: $8 + d \times 8$ bytes.
   - For n sketches: $n \times (8 + d \times 8)$.

Combining these components:

- Array of filenames: $n \times 8$ bytes.
- Array of `Sketch` objects: $n \times 8$ bytes.
- Total for `Sketch` objects: $n \times (8 + d \times 8)$ bytes.

Thus, the total memory usage M as a function of n and d is:

$$M(n, d) = n \times 8 + n \times 8 + n \times (8 + d \times 8)\, M(n, d) = n \times (8 + 8 + 8 + d \times 8)$$
$$M(n, d) = n \times (24 + 8d)$$

So, the estimated memory usage is $\mathbf{M(n, d) = n \times (24 + 8d)}$ bytes.